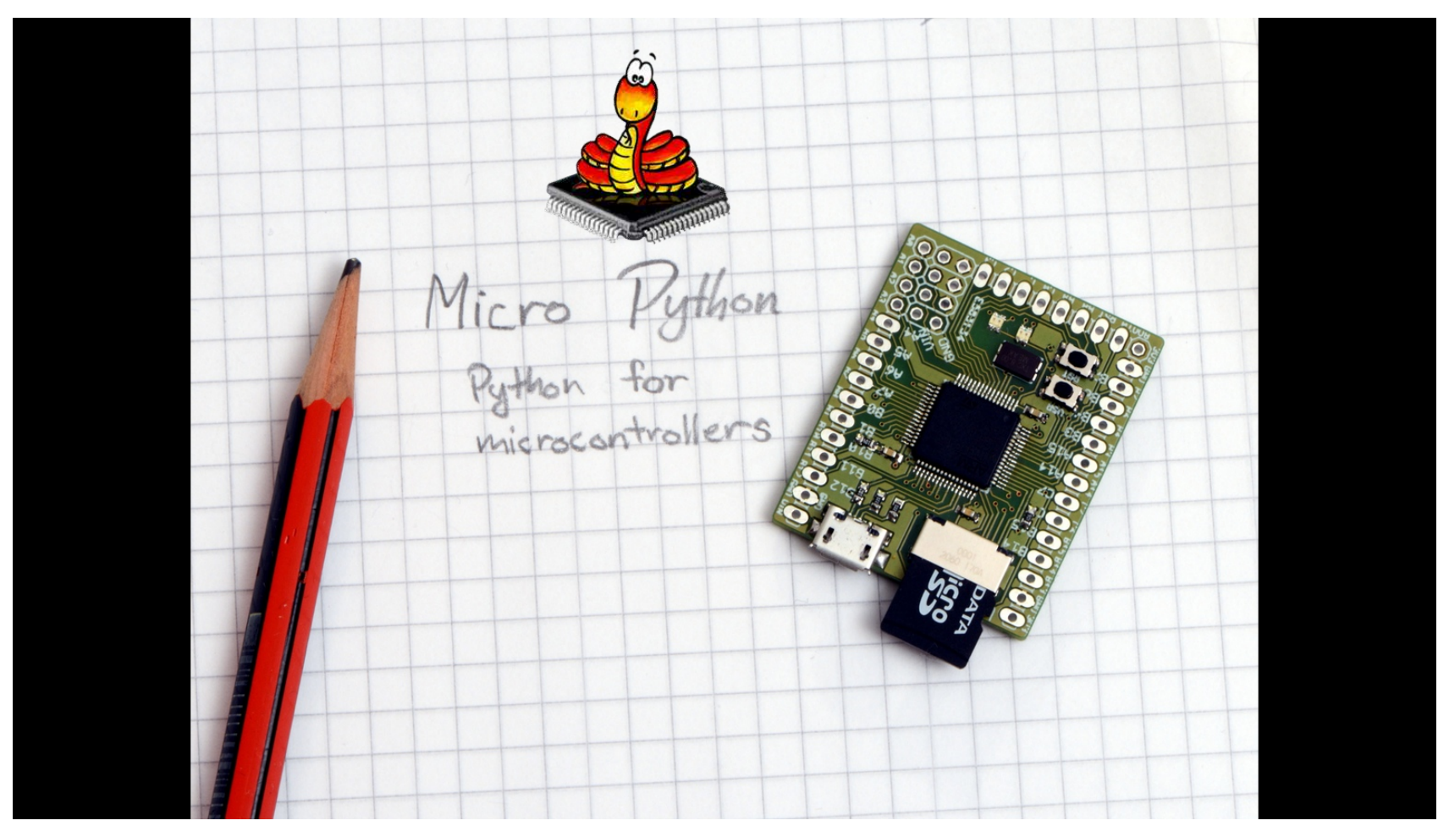
MicroPython – Python for Microcontrollers



Micro Python is a complete rewrite, from scratch, of the Python scripting language. It is written in clean, ANSI C and includes a complete parser, compiler, virtual machine, runtime system, garbage collector and support libraries to run on a microcontroller.
The compiler can compile to byte code or native machine code, selectable per function using a function decorator. It also supports inline assembler.
All compilation happens on the chip, so there is no need for any software on your PC.

Proper Python with hardware-specific modules

MicroPython is a full Python compiler and runtime that runs on the bare-metal. You get an interactive prompt (the REPL) to execute commands immediately, along with the ability to run and import scripts from the built-in filesystem.
The REPL has history, tab completion, auto-indent and paste mode for a great user experience.

MicroPython strives to be as compatible as possible with normal Python (known as CPython) so that if you know Python you already know MicroPython. On the other hand, the more you learn about MicroPython the better you become at Python.

Code: state-of-the-art and highly robust

MicroPython employs many advanced coding techniques, and lots of tricks to maintain a compact size while still having a full set of features.

- highly configurable due to many compile-time configuration options
- support for many architectures (x86, x86-64, ARM, ARM Thumb, Xtensa)

- extensive test suite with over 590 tests, and more than 18,500 individual testcases
- code coverage at 98.4% for the core and at 96.3% for the core plus extended modules

- fast start-up time from boot to loading of first script (150 microseconds to get to boot.py, on PYBv1.1 running at 168MHz)
- a simple, fast and robust mark-sweep garbage collector for heap memory

- a MemoryError exception is raised if the heap is exhausted
- a RuntimeError exception is raised if the stack limit is reached

- support for running Python code on a hard interrupt with minimal latency
- errors have a backtrace and report the line number of the source code

- constant folding in the parser/compiler
- pointer tagging to fit small integers, strings and objects in a machine word

- transparent transition from small integers to big integers
- support for 64-bit NaN boxing object model

- support for 30-bit stuffed floats, which don't require heap memory
- a cross-compiler and frozen bytecode, to have pre-compiled scripts that don't take any RAM (except for any dynamic objects they create)

- multithreading via the "_thread" module, with an optional global-interpreter-lock (still work in progress, only available on selected ports)
- a native emitter that targets machine code directly rather than the bytecode virtual machine
- inline assembler (currently Thumb and Xtensa instruction sets only)

Is MicroPython fully comparable to Python 3.4?

MicroPython has exactly the same grammar (syntax) as Python version 3. This means that the way you write Python code is exactly the same in MicroPython (for loops, function definitions, classes, list comprehension, etc). Scripts that compile in normal Python will compile in MicroPython, and vice versa.

MicroPython does not at the moment implement all of Python's standard libraries. Some Python standard libraries are written in C and need to be re-written to work with MicroPython. Ultimately, not all Python libraries will be fully supported because they are not feasible to run on a microcontroller, either because the functionality is not available on the microcontroller, or because they take too much memory.

Ranking in the top 100 of the most popular projects on GitHub in C/C++

More than 160 contributors, 4 850 stars and 1000 forks on GitHub
See for yourself: https://github.com/micropython

Completely free Open Source and MIT license

MicroPython is written in C99 and the entire MicroPython core is available for general use under the very liberal MIT license. Most libraries and extension modules (some of which are from a third party) are also available under MIT or similar licenses.

You can freely use and adapt MicroPython for personal use, in education, and in commercial products.

MicroPython is developed in the open on GitHub and the source code is available at the GitHub page, and on the download page. Everyone is welcome to contribute to the project.

Hardware that runs MicroPython

- George Robotics:
  MicroPython pyboard and MicroPython pyboard lite
- Adafruit: Feather M0 Express
- BBC:MicroBit
- Digi: XBee Cellular LTE Cat 1
- OpenMV: Cam M7 and Cam M4
- ST:
  WiFi SPWF04

  NUCLEO-F401RE
  NUCLEO-F429ZI
  NUCLEO-F446EI
  NUCLEO-F767ZI

  STM32F429I-DISCO
  STM32F746G-DISCO

MicroPython in the press

"MicroPython: more powerful than Arduino, simpler than the Raspberry Pi"
2013 WIRED UK

MicroPython was announced as the "Linux for the IoT"
2016 Elektor Magazine

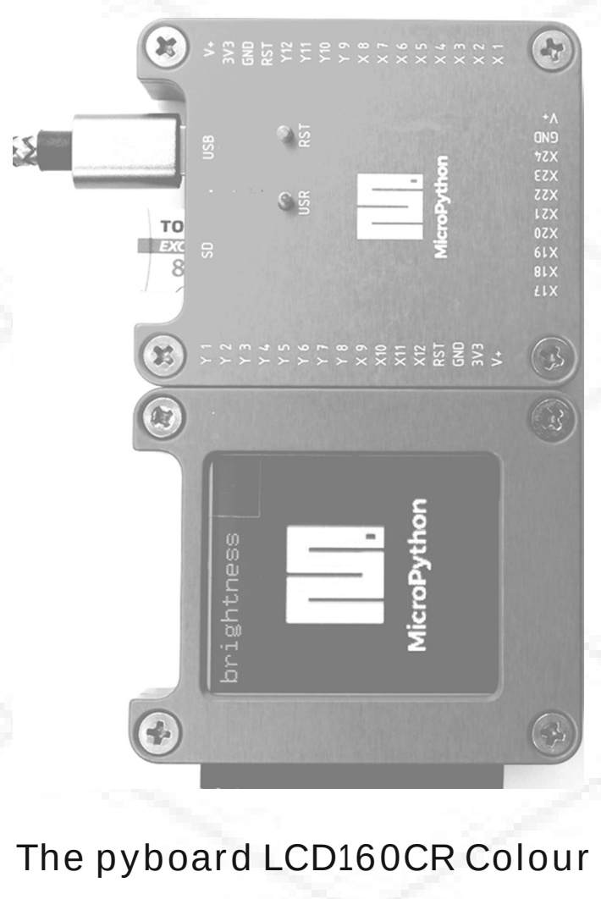
Who is Damien George?

Damien was born in Melbourne, Australia, and has been programming and playing with electronic circuits since primary school.
He completed a Bachelor of Engineering and Bachelor of Science at the University of Melbourne, and then went on to complete a PhD in theoretical physics.

During his studies he participated in the international Robocup competition, programming autonomous robots to play soccer. He wrote embedded software for scripted behavioural control and motion, as well as building parts of the hardware. He has since continued in this area, building robots, a CNC machine, and writing embedded software for many microcontrollers.

He worked professionally as a theoretical physicist for 6 years, on various topics including cosmology and the Higgs boson.
He then went on to develop MicroPython and ran two very successful Kickstarter campaigns around this microcontroller language.
He now works full-time maintaining the MicroPython code-base and ecosystem.

The Evolution of MicroPython



The MicroPython pyboard

The MicroPython pyboard is a compact electronic circuit board that runs MicroPython on the bare metal, giving you a low-level Python operating system that can be used to control all kinds of electronic projects.

MicroPython is packed full of advanced features such as an interactive prompt, arbitrary precision integers, closures, list comprehension, generators, exception handling and more. Yet it is compact enough to fit and run within just 256k of code space and 16k of RAM.

MicroPython aims to be as compatible with normal Python as possible to allow you to transfer code with ease from the desktop to a microcontroller or embedded system.

The pyboard LCD160CR Colour Display with resistive touch

Powered by the original pyboard v1.1 for you to play with.
You can connect the Display to the left or the right side next to the pyboard.

Touch the display to see the next demo and how intensely the CPU is used for this demo.
How many frames per second are possible is shown as well.

- controlled with intelligence and optimised for Python programming
- integrated touch controller
- low RAM footprint
- heapless library is integrated in MicroPython

http://micropython.org/unicorn/

Use MicroPython online:
Write a script, paste some code or try a demo!

2018

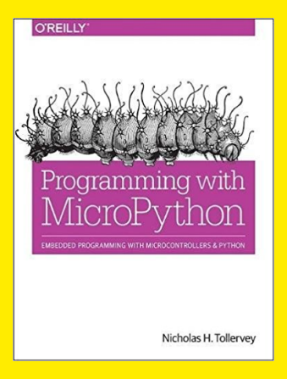
E-Paper skin for the pyboard

OLED skin for the pyboard – you can see first prototypes here:

Introduction of a modular system to easily build electronic devices

2017

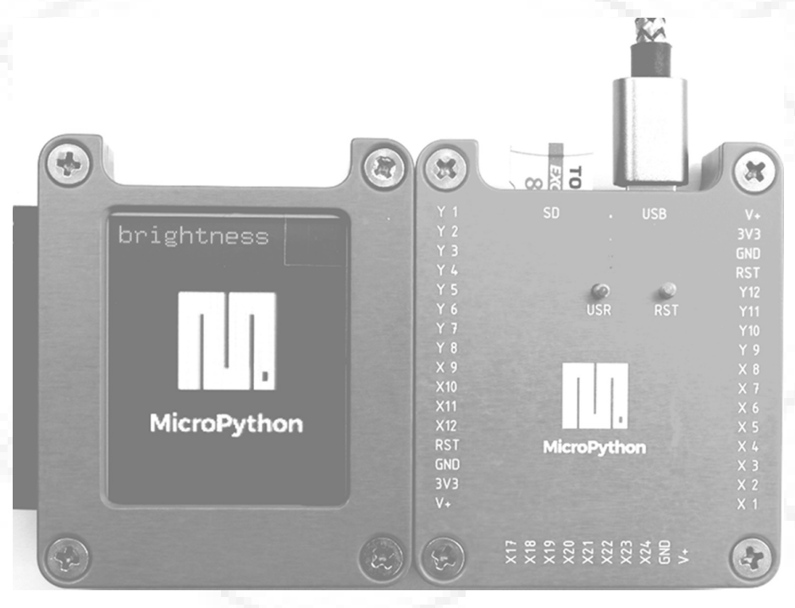
E-Agle Trento Racing Team uses the pyboards for controlling the interface for the driver of a racing car.



Learn how to use MicroPython to make cool stuff. This practical book assumes no previous knowledge of programming and takes you on a journey from first steps to advanced projects. Written by the programmer who proposed, coordinated, and contributed to getting MicroPython on the BBC micro: bit, there's no better person to teach you this topic.

MicroPython Colour OLED

- Colour OLED with optimised interface for all pyboards
- same MicroPython experience for Low Power applications
- low RAM footprint
- heapless library is integrated in MicroPython
- four Buttons
- power optimised
- optimised for python programming
- dynamic power consumption
- no background illumination
- power scales with number of bright pixels
- power consumption 10 mA to 100 mA (max brightness)



The MicroPython pyboard lite

For low power applications in the IoT world George Robotics developed the MicroPython pyboard lite.
Having a board full compartible with the existing high performance pyboard.

2016

MicroPython went to school

2 Talks at the PyCon Australia:

From Kickstarter to Space
Scripting the Internet of Things

2nd Kickstarter: MicroPython on the ESP8266
MicroPython run like clockwork on the bare metal for ESP8266 Wi-Fi

March 2016: 1384 backers raised 28 334 GBP pure Software Campagne with multiple streach goals

- develop, test and document a suite of drivers for environmental sensors that can run on the ESP8266 (and other MP boards).
- Implement a simple micro database on the ESP8266.
- Implement a native emitter for the ESP8266 (Xtensa architecture).

NEW LOGO

How MicroPython was ported to the BBC Micro:Bit by Nicholas N. Tollervey

In 2015 the BBC explained a staggering "moon-shot" project: to create a small, computing device to be delivered to ALL year 7 children (11-12 years old) in the UK. The newly christened BBC micro:bit would facilitate the first step towards inspiring digital creativity in a new generation of school children. Nicholas Torvalls brought the BBC:MicroBit and Damien George together. Besides a couple of other programming languages it's now possible to run MicroPython on this device.

please visit: http://ntoll.org/article/story-micropython-on-microbit to read the full story.

Prestigious Community Service Award by Python Software Foundation

For his extensive volunteer work on the BBC micro:bit and MicroPython Damien Geoorge received the prestigious Community Service Award. Furthermore, Damien has spent time answering questions, offering help and reviewing code from the wider MicroPython/micro:bit community. Through his work on the MicroPython board, optimizations have also been made to CPython's speed. https://www.python.org/community/awards/psf-awards/...

George Robotics Limited (the company behind MicroPython) is proud to announce that the European Space Agency (ESA) will be funding further development of MicroPython, to determine the suitability of the language for space-based applications, in particular for payloads.

Research and development will focus on making MicroPython more robust for critical embedded systems, with emphasis on determinism of the virtual machine and memory management.

The research program foresees the development of a port of MicroPython to the SPARC architecture, which will be made available under an ESA community license.

ESA has generously agreed that all improvements to MicroPython that are made as part of this R&D program can be incorporated into the generic implementation.

2015

MicroPython went to space

Pyboard in a rocket!
(Rocket Launch for International Student Satellites (ARLISS)
A pyboard was used to meassure acceleration in the rocket and it went up to 12,000 ft

Port to LEON/SPARC/RTEMS for Space

separation of the VM and compiler
cross compiler and persistent bytecode
64-bit NaN-boxing object model
understandong of determinism

Satellite Control at the application layer

http://micropython.org/live/

A MicroPython pyboard is connected to the internet for you to play with!
Scan the QR-code and see for yourself

2014

MicroPython pyboard available for everyone

2 Talks at the PyCon UK:

Micro Python - Kickstarter Experience
Micro Python - shrinking Python down to run on a microcontroller

George Robotics Ltd – the company behind MicroPython was founded to continue the development of MicroPython

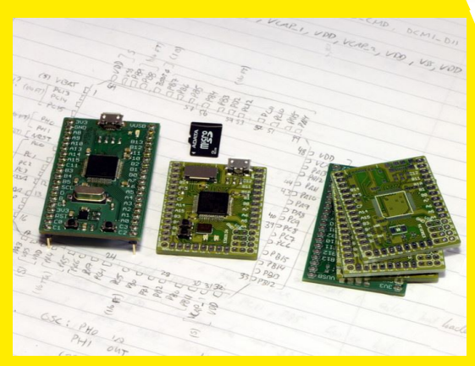
2013

Is it possible to put Python on a Microcontroller?

April 2013: The idea for MicroPython

September 2013: first flashing LED in MicroPython
October 2013: REPL and filesystem

Kickstarter Launch at 13 November 2013 with funding goal 15 000 GBP.
After 30 days 1931 backers raised 97 803 GBP to make this project happen



Streach Goals:

- 40 000 GBP Wi-Fi support via CC3000 module
- 50 000 GBP Ethernet WIZ820io
- 60 000 GBP NRF24L01 low power wireless module

```
MicroPython has an inline assembler

import micropython

# define a Thumb-code inline-assembler function
@micropython.asm_thumb
def asm_add(r0, r1):
   add(r0, r0, r1)

# use it as a normal Python function
total = asm_add(1, 2)
```

```
full range of numeric types

# small integer (fits in a machine word)
>>> 123
123

# big integer
>>> 1 << 160
1461501637330902918203684832716283019655932542976

# floating point
>>> 1.23e6
1230000.0

# complex numbers
>>> (1 + 2j) * 4j
(-8+4j)
```

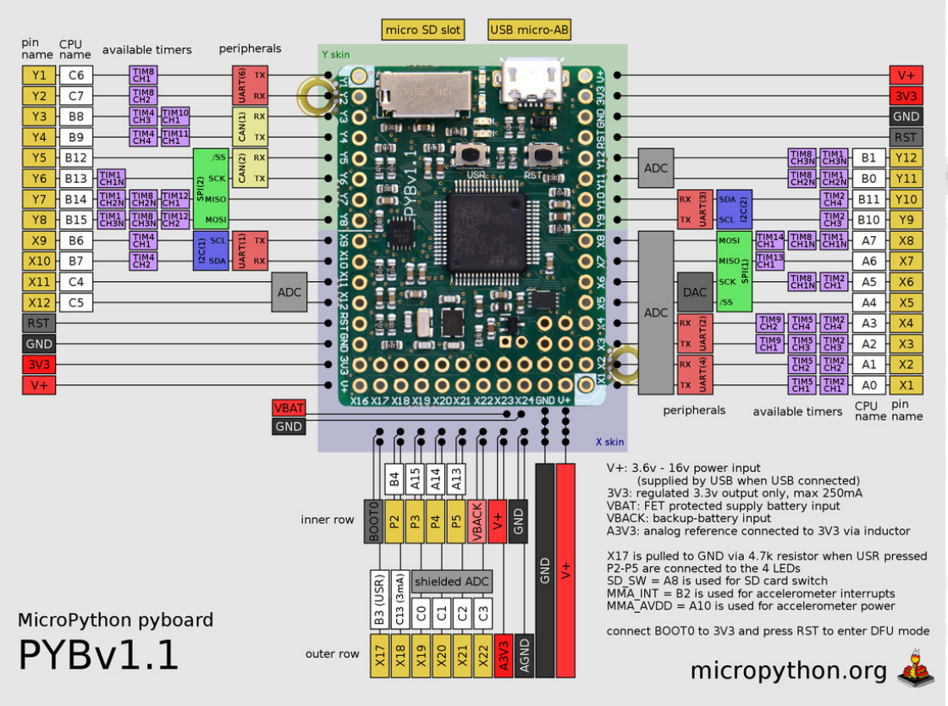
```
import time

time.sleep(1)           # sleep for 1 second
time.sleep_ms(500)      # sleep for 500 milliseconds
time.sleep_us(10)       # sleep for 10 microseconds
start = time.ticks_ms() # get millisecond counter
delta = time.ticks_diff(time.ticks_ms(), start) # compute time difference
```

```
DAC (digital to analog conversion)

from pyb import Pin, ADC

# read value, 0-4095
adc = ADC(Pin('X19'))
adc.read()
```


micro SD slot
USB micro-AB
pin name
CPU name
available timers
peripherals
Y skin
X skin
inner row
outer row
shielded ADC
V+: 3.6v - 16v power input (supplied by USB when USB connected)
3V3: regulated 3.3v output only, max 250mA
VBAT: FET protected supply battery input
VBACK: backup-battery input
A3V3: analog reference connected to 3V3 via inductor
X17 is pulled to GND via 4.7k resistor when USR pressed
P2-P5 are connected to the 4 LEDs
SD_SW = A8 is used for SD card switch
MMA_INT = B2 is used for accelerometer interrupts
MMA_AVDD = A10 is used for accelerometer power
connect BOOT0 to 3V3 and press RST to enter DFU mode
MicroPython pyboard
PYBv1.1
micropython.org

```
MicroPython has a file system

import os

# list root directory
print(os.listdir('/'))

# print current directory
print(os.getcwd())

# open and read a file from the SD card
with open('/sd/readme.txt') as f:
   print(f.read())
```

```
import pyb

# duplicate REPL on UART(1)
pyb.repl_uart(pyb.UART(1, 9600))
# pause CPU, waiting for interrupt
pyb.wfi()
# get CPU and bus frequencies
pyb.freq()
# set CPU freq to 60MHz
pyb.freq(60000000)
# stop CPU, waiting for external interrupt
pyb.stop()
```

More information at www.micropython.org